

Testing Top Monotonicity

Haris Aziz

NICTA and UNSW Australia, Kensington 2033, Australia

Abstract

Top monotonicity is a relaxation of various well-known domain restrictions such as *single-peaked* and *single-crossing* for which negative impossibility results are circumvented and for which the *median-voter theorem* still holds. We present a polynomial-time algorithm to test whether a given preference profile satisfies top monotonicity or not.

Keywords: Social choice theory, domain restrictions, top monotonicity, single peaked, single crossing, computational complexity.

JEL: C63, C70, C71, and C78

April 1, 2014

1. Introduction

The standard social choice setting is one in which a set of agents N express preferences over a set of alternatives A and the goal is to select an alternative based on the preferences. Social choice theory is replete with results concerning the impossibility of the voting rules simultaneously satisfying desirable axioms. Prominent among these results are *Condorcet's Paradox* [14], *Arrow's Theorem* [1], and the *Gibbard Satterthwaite Theorem* [15]. One approach to circumvent such results is identifying restricted domains in which impossibility results disappear. Examples of domain restrictions include preference profiles that are *single-peaked*; *single-plateaued* [20] *single crossing* [21]; or *order restricted* [13]. For example, the Condorcet paradox does not occur when preferences are single-peaked. The *median voter theorem* states that for single-peaked preference profiles, Condorcet winner(s) exist and they coincide with the median(s) of the voters' most preferred alternatives [6].

Restricting domains to test robustness of impossibility results, and to identify settings that admit attractive preference aggregation rules, has been a fruitful area of research within social choice [3]. Whereas economists have examined axiomatic implications of domain restrictions [see e.g., 23, 17, 16, 22, 20], computer scientists are starting to examine natural problems such as checking whether a preference profile satisfies a certain structural property. On the computational side, Bartholdi, III and Trick [5] and Escoffier et al. [12] showed

that checking whether a profile is single-peaked is polynomial-time solvable. Elkind et al. [10] and Brederbeck et al. [9] then proved that it can be checked in polynomial time whether a profile is single-crossing. There is related work on checking whether *other* computational problems such as manipulation become easy when the preference profile satisfies some structure [see e.g., 7]. There has also been some work on identifying ‘almost’ nicely structured profiles [see e.g., 8, 11]. However, we are interested in the more fundamental problem of *checking* whether a preference profile satisfies a given structural property.

Whereas the domain restrictions listed above have garnered a lot of attention, axiomatic results for these restrictions have been somewhat piecemeal. Recently, Barberà and Moreno [4] proposed a new consistency condition called *top monotonicity* that is a relaxation of all the domain restrictions listed above but which still is a sufficient condition for an extension of the median voter theorem to hold. Top monotonicity also allows for indifferences both among the maximally preferred alternatives and non-maximally preferred alternatives. Barberà and Moreno [4] write that “*Among other things, top monotonicity will stretch the extent to which one may accommodate indifferences and still obtain positive results regarding Condorcet winners in the case of the majority rule, or of voting equilibria, more generally.*” Top monotonicity is also induced in two natural of tax rate determination [Appendix A, 4].

Although the axiomatic aspects of top monotonicity have been studied [3, 4], it is not clear how easy it is to test top monotonicity of a preference profile. Barberà and Moreno [4] ask that “*Is the satisfaction of our conditions easy to check? Top monotonicity may be easy to check for in some cases, and also easy to discard, in others.*”. In view of the generality of top-monotonicity, similar computational investigations of restricted domains [5, 12, 18, 2], and the questions raised by Barberà and Moreno [4], we study the following natural problem:

What is the computational complexity of checking whether a profile satisfies top-monotonicity?

We show that although the straightforward algorithm to test top monotonicity requires going through $|A|!$ orders, there exists a polynomial-time algorithm to test top monotonicity. The algorithm relies on our characterization of top monotonicity with respect to certain violation constraints.

2. Top Monotonicity

Consider the social choice setting in which there is a set of agents $N = \{1, \dots, n\}$, a set of alternatives $A = \{a_1, \dots, a_m\}$ and a preference profile $\succsim = (\succsim_1, \dots, \succsim_n)$ such that each \succsim_i is a complete and transitive relation over A . We write $a \succsim_i b$ to denote that agent i values alternative a at least as much as alternative b and use \succ_i for the strict part of \succsim_i , i.e., $a \succ_i b$ iff $a \succsim_i b$ but not $b \succsim_i a$. Finally, \sim_i denotes i ’s indifference relation, i.e., $a \sim_i b$ iff both $a \succsim_i b$ and $b \succsim_i a$.

We define by $t_i(S)$ the set of maximal elements \succsim_i on S . Let $A(\succsim)$ be the family of sets containing A itself, and also all triples of distinct alternatives where each alternative is top on A for some agent $k \in N$ according to \succsim . We are now in a position to present the definition of top monotonicity.

Definition 1 (Top monotonic). *A preference profile \succsim is top monotonic iff there exists a linear order $>$ over A such that*

- (i) $t_i(A)$ is a finite union of closed intervals for all $i \in N$, and
- (ii) For all $S \in A(\succsim)$, for all $i, j \in N$, all $x \in t_i(S)$, all $y \in t_j(S)$, and any $z \in S$, the following holds:

$$[x > y > z \text{ or } z > y > x] \implies$$

$$[y \succsim_i z \text{ if } z \in t_i(S) \cup t_j(S) \text{ and } y \succ_i z \text{ if } z \notin t_i(S) \cup t_j(S)]. \quad (1)$$

We will say that \succsim satisfies top monotonicity with respect to order $>$.

Remark 1. *Top monotonicity does not preclude cycles in majority comparisons, but only guarantees that these do not occur at the top of the majority relation.*

Remark 2. *Top monotonicity on a set of alternatives is not necessarily inherited on its subsets.*

3. Testing top monotonicity

In this section, we examine the problem of testing top monotonicity.

3.1. Initial observations

It follows from Definition 1, that top monotonicity of a profile with respect to a particular order can be checked easily.

Observation 1. *For a given order over the alternatives, it can be checked in $O(|A|^3 \cdot |N|^3)$ whether the preference profile satisfies top monotonicity with respect to that order.*

In view of the observation above, one can straightforwardly design an algorithm to test top monotonicity: go through each of the $|A|!$ orders and checks whether top monotonicity is satisfied with respect to one of the orders.

Observation 2. *It can be checked in time $O(|A|^3 \cdot |N|^3 \cdot |A|!)$ whether the preference profile satisfies top monotonicity.*

A natural question is whether it is possible to test top monotonicity *without* having to go through all the $|A|!$ orders. First, we make another observation.

Lemma 1. *If a preference profile satisfies top monotonicity with respect to an order, it also does so with respect to the reverse of the order.*

Proof. Note that for any $S \in A(\succsim)$, and for each $i, j \in N$, each $x \in t_i(S)$, $y \in t_j(S)$, and $z \in S$, the conditions of (i) and (ii) in the definition of top monotonicity are not affected if $>$ is reversed. \square

3.2. Non-betweenness violation constraints

In the rest of the section, we will heavily use the idea of *NB* (*non-betweenness*) violation constraints on the prospective order on A . We now introduce these constraints.

Definition 2 (Non-betweenness violation constraint). *For $x, y, z \in A$, we call $a(y, \{x, z\})$ a NB (non-betweenness) violation constraint for an order $>$ over A if neither $x > y > z$ nor $z > y > x$ can hold.*

Next, we show that the existence of an order on A satisfying an *NB* constraint set C can be checked efficiently.

Lemma 2. *For an NB constraint set C on alternative set A , there exists a $O(|C| \cdot |A|)$ algorithm to check whether constraints in C can be satisfied by some partial order.*

Proof. The high-level idea of the algorithm is to iteratively go through the constraints in C and refine a partial order P accordingly. Initially P is set to the empty relation so that elements of A are not comparable to each other. Based on each *NB* constraint, P is refined and more arcs are added between alternatives in A . Each time we add arcs we also add arcs that respect the transitive closure. For some cases, two distinct clones of alternatives are maintained to indicate two possible relative positions of that alternative. If all the constraints are dealt with and the partial order P is still feasible then we return P . Otherwise we return no.

For the first constraint $(x, \{y, z\})$, we simply put x above y and z . For each constraint $(x, \{y, z\})$ considered, either we add arcs to/from a maximum of two ‘clones’ each of x, y, z , or eliminate one of the clones of x, y, z , or arrive at a contradiction so that no partial order P and hence no linear order $>$ satisfies the constraints in C .

Now we give details as how each constraint $(a, \{b, c\}) \in C$ is used to modify the partial order P . The partial order is represented by a graph in which all arcs go downwards. $a \longrightarrow \{b, c\}$ represents $a P \{b, c\}$ i.e., both $a P b$ and $a P c$. Each time we consider a new constraint $(a, \{b, c\})$ we do the following:

- If a is not comparable to b or c with respect to P i.e., if the relative position of a with respect to $\{b, c\}$ is not confirmed, then we create two clones of a : a_1 and a_2 to represent the two possible relative positions of alternative a : $a_1 P \{b, c\}$ or $\{b, c\} P a_2$. Thus we put a_1 above $\{b, c\}$ and a_2 below $\{b, c\}$ with arcs added accordingly.
- If $b P a$ or $c P a$, then a is placed below $\{b, c\}$ with arcs going from b and c to a .
- If $a P b$ or $a P c$, then a is placed above $\{b, c\}$ with arcs going from a to b and c .

- If we already have that $b P a P c$ or $c P a P b$ then we know that no partial order can satisfy all the constraints of C and we can already return no. In some cases we do not return no immediately but remove from contention one of the clones of the alternative. For example, if $b P a_2 P c$, then a_2 is deleted. If a_1 has also been deleted then we return no. Otherwise a_1 is the only plausible clone of a left for which a partial order may satisfy the constraints in C . Thus we return no if both clones of an alternative need to be deleted for the constraint to be satisfied.

For each constraint, we require a maximum of $|A|$ operations on the partial order. Note that we do not have to create more than two clones of any alternative. If one of the clones is in the correct relative position, then it should itself satisfy all the constraints pertaining to it. Hence, we just need to restrict our attention to the two clones that are initially formed in the partial order constructed. \square

Lemma 3. *For an NB constraint set C on alternative set A , there exists a partial order that satisfies the constraints in C if and only if there exists a linear order that satisfies the constraints in C*

Proof. (\Rightarrow) If a partial order satisfies the constraints in C , then the arcs in the graph representing the partial order already capture all the non-betweenness constraints. Any additional arcs do not negate these constraints. Hence, any linear extension of the partial order also satisfies the constraints in C .

(\Leftarrow) This direction is trivial since a linear order is also a partial order. \square

Lemma 4. *For an NB constraint set C on alternative set A , there exists a $O(|C| \cdot |A|)$ algorithm to check whether constraints in C can be satisfied by some linear order.*

Proof. The statement follows from Lemmas 2 and 3. \square

Example 1 illustrates the algorithm for checking whether NB constraints can be satisfied by some linear order or not.

Example 1 (Illustrating the algorithm in the proof of Lemma 2). *Let us say that $C = \{(y, \{x, z\}), (y', \{x, z\}), (x, \{z, y\}), (x, \{y, y'\})\}$ and we deal with constraints from left to right.*

- (i) $(y, \{x, z\})$: y is ordered above $\{x, z\}$ because x, y, z was the first triple we considered (Figure 1).
- (ii) $(y', \{x, z\})$: y' is in consideration for both above of $\{x, z\}$ and below of $\{x, z\}$ (Figure 2).
- (iii) $(x, \{z, y\})$: x is shifted below of y (Figure 3).
- (iv) $(x, \{y, y'\})$: the second (lower) clone y'_2 of y' is deleted (Figure 4).

In that case the relative ordering has to be $\{y, y'\} P z P x$ where the relative ordering of y and y' is not yet fixed. Hence both of the following orders are valid linear extensions of P : $y > y' > z > x$ and $y > y > z > x$.

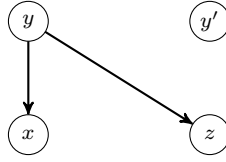


Figure 1: Stage 1 — implementing constraint $(y, \{x, z\})$ in Example 1.

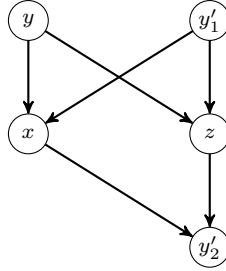


Figure 2: Stage 2 — implementing constraint $(y', \{x, z\})$ in Example 1.

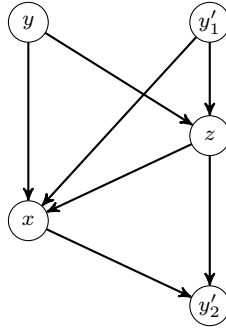


Figure 3: Stage 3 — implementing constraint $(x, \{z, y\})$ in Example 1.

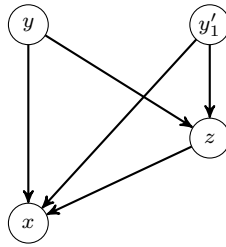


Figure 4: Stage 4 — implementing constraint $(x, \{y, y'\})$ in Example 1.

3.3. Algorithm to test top monotonicity

In the next two lemmas, we show how NB constraints can be used to capture top monotonicity.

Lemma 5. *For an order $>$, and $i \in N$, $t_i(A)$ is a closed interval iff for any $x, z \in t_i(A)$ and any $y \notin t_i(A)$, the NB constraint $(y, \{x, z\})$ holds.*

Proof. Case (i) requires that for any $x, z \in t_i(A)$ and any $y \notin t_i(A)$, y is not between x and z . Hence $(y, \{x, z\})$ has to hold. \square

Lemma 6. *For an order $>$, case (ii) in definition of top monotonicity is always satisfied iff for all $S \in A(\succsim)$, for all $i, j \in N$, all $x \in t_i(S)$, all $y \in t_j(S)$, and any $z \in S$, the NB constraint $(y, \{x, z\})$ holds.*

Proof. For each $S \in A(\succsim)$, and for each $i, j \in N$, each $x \in t_i(S)$, $y \in t_j(S)$, and $z \in S$, we check which of the following relative orders do not lead to a lack of top monotonicity. The following relative orders over x, y, z trivially satisfy the top monotonicity condition (ii) for set S and voter i because they do not feature in the antecedent of implication (1).

$$(i) \ y > x > z$$

$$(ii) \ y > z > x$$

$$(iii) \ x > z > y$$

$$(iv) \ z > y > x$$

The following relative orders in which y is in between x and z are only in contention if the consequent of implication (1) holds.

$$(i) \ x > y > z$$

$$(ii) \ z > y > x$$

If the consequent holds, then any of the six relative orders are possible and there is no constraint for this 3-set of alternatives. If the consequent does not hold, then we can safely say that both x and z are on one side of y i.e., y is cannot be between x and z . Thus in this case we have a constraint on the ordering over A that y cannot be between x and z . Therefore for each 3-set of alternatives, either there is no constraint or there is a constraint $(y, \{x, z\})$: that y cannot be between x and z . Note that $(y, \{x, z\})$ is a required constraint if even for one i, j and S , the consequent of implication (1) does not hold. \square

Next, we argue that the top monotonicity can be tested in polynomial time. The main idea is to reduce the problem to check whether there exists a linear order that satisfies a carefully constructed set of NB constraints.

Theorem 1. *Top monotonicity can be checked in polynomial time $O(|N|^2 \cdot |A|^6)$.*

Proof. In the algorithm, we first construct a set C of NB constraints that an order must satisfy. Firstly we put in C , all NB constraints pertaining to Lemma 5. Secondly, we put in C , NB constraints pertaining to case (ii) of the definition of top monotonicity as follows. For each $S \in A(\succsim)$, and for each

$i, j \in N$, each $x \in t_i(S)$, $y \in t_j(S)$, and $z \in S$, we check whether the consequent of the implication (1) holds. If it does not hold then we add $(y, \{x, z\})$ to C . Note that $|C| \leq |A|^3$ and C can be constructed in time $O(|N|^2 \cdot |A|^3 \cdot |A|^3)$.

Now that we have constructed the constraint set C , we can use the algorithm in the proof of Lemma 2 to check whether there exists an order $>$ on A that satisfies the NB constraints in C . The algorithm takes time $O(|C| \cdot |A|) = O(|A|^4)$. The total running time is then $O(|N|^2 \cdot |A|^3 \cdot |A|^3) + O(|A|^4) = O(|N|^2 \cdot |A|^6)$. It follows from Lemmas 5 and 6, that profile \succsim satisfies top monotonicity with respect to order $>$ iff $>$ satisfies the constraints in C . Hence profile \succsim is top monotonic iff there exists a partial order P that satisfies all the constraints in the constructed constraint set C . By Lemma 1, the partial order constructed can also be reversed and still satisfies top monotonicity. \square

Barberà and Moreno [4] defined *peak monotonic* preferences as top monotonic preferences in which $|t_i(A)| = 1$ for all $i \in N$. Hence our proposed algorithm can also be used to test peak monotonicity. The algorithm can also help to identify ‘no’ instances for special cases of top monotonicity. For example if the algorithm returns ‘no’ for a particular preference profile, we immediately know that the profile is also not single-peaked or single-crossing. The general technique proposed to check for top monotonicity by appealing to non-betweenness constraints (Lemma 2) may also be useful to test other domain restrictions.

For future work, we can use the algorithm to check whether real world preferences (such as in prefib [19]) exhibit top monotonicity. Due to the generality of top monotonicity, it is plausible that it is much more wide-spread than single-peakedness. Finally, the complexity of profiles that are ‘almost top monotonic’ may also be interesting.

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. The author thanks Bernardo Moreno Jimenez for some helpful pointers.

References

- [1] Arrow, K. J., 1963. Social Choice and Individual Values, 2nd Edition. New Haven: Cowles Foundation, 1st edition 1951.
- [2] Aziz, H., Brill, M., Harrenstein, P., 2013. Testing substitutability of weak preferences. Mathematical Social Sciences 66 (1), 91–94.
- [3] Barberà, S., Berga, D., Moreno, B., 2013. Some new domain restrictions in social choice, and their consequences. In: Modeling Decisions for Artificial Intelligence. pp. 11–24.

- [4] Barberà, S., Moreno, B., 2011. Top monotonicity: A common root for single peakedness, single crossing and the median voter result. *Games and Economic Behavior* 73 (2), 345–359.
- [5] Bartholdi, III, J., Trick, M., 1986. Stable matching with preferences derived from a psychological model. *Operations Research Letters* 5 (4), 165–169.
- [6] Black, D., 1948. On the rationale of group decision-making. *Journal of Political Economy* 56 (1), 23–34.
- [7] Brandt, F., Brill, M., Hemaspaandra, E., Hemaspaandra, L., 2010. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In: Fox, M., Poole, D. (Eds.), *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, pp. 715–722.
- [8] Brederbeck, R., Chen, J., Woeginger, G. J., 2013. Are there any nicely structured preference profiles nearby? In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*. pp. 62–68.
- [9] Brederbeck, R., Chen, J., Woeginger, G. J., 2013. A characterization of the single-crossing domain. *Social Choice and Welfare* 41 (1), 989–998.
- [10] Elkind, E., Faliszewski, P., Slinko, A., 2012. Clone structures in voters’ preferences. In: *Proceedings of the 13th ACM Conference on Electronic Commerce (ACM-EC)*. pp. 496–513.
- [11] Erdélyi, G., Lackner, M., Pfandler, A., 2013. Computational aspects of nearly single-peaked electorates. In: *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, pp. 283–289.
- [12] Escoffier, B., Lang, J., Öztürk, M., 2008. Single-peaked consistency and its complexity. In: *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*. IOS Press, pp. 366–370.
- [13] Gans, J. S., Smart, M., 1996. Majority voting with single-crossing preferences. *Journal of Public Economics* 59 (2), 219–237.
- [14] Gehrlein, W. V., 2006. *Condorcet’s Paradox*. Springer-Verlag.
- [15] Gibbard, A., 1973. Manipulation of voting schemes. *Econometrica* 41, 587–602.
- [16] Inada, K., 1964. A note on the simple majority decision rule. *Econometrica* 32 (4), 525–531.
- [17] Inada, K., 1969. The simple majority decision rule. *Econometrica* 37, 490–506.
- [18] Knoblauch, V., 2010. Recognizing one-dimensional Euclidean preference profiles. *Journal of Mathematical Economics* 46 (1), 1–5.

- [19] Mattei, N., Walsh, T., 2013. PrefLib: A library for preference data <http://www.preflib.org>. In: Perny, P., Pirlot, M., Tsoukiàs, A. (Eds.), Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT). Vol. 8176 of Lecture Notes in Computer Science (LNCS). Springer, pp. 259–270.
- [20] Moulin, H., 1980. On strategy-proofness and single peakedness. *Public Choice* 35 (4), 437–455.
- [21] Roberts, K. W. S., 1977. Voting over income tax schedules. *Journal of Public Economics* 8 (3), 329–342.
- [22] Rothstein, P., 1990. Order restricted preferences and majority rule. *Social Choice and Welfare* 7 (4), 331–342.
- [23] Sen, A. K., Pattanaik, P. K., 1969. Necessary and sufficient conditions for rational choice under majority decision. *Journal of Economic Theory* 1, 178–202.